6. Scheduler

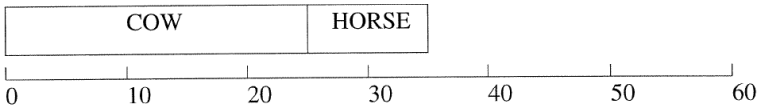
Program Name: Scheduler.java Input File: scheduler.dat

Real-time systems are systems that are subject to a "real-time constraint". Essentially, real-time programs must guarantee response within specified time constraints, or "deadlines". Periodic real-time systems are ones in which the programs have periodic tasks. If a set of periodic tasks can be scheduled in such a way that their constraints are always met, then that set of tasks is called "schedulable".

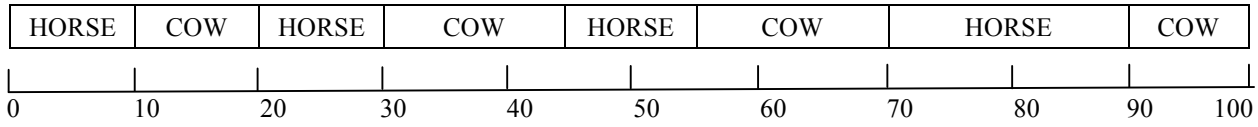
Given the periodic requirements, it is trivial to show that the tasks are schedulable if and only if there exists a valid scheduling for the first N minutes, where N is the LCM (least common multiple) of the periods.

For an example of a periodic system, consider one in which you are a rancher and you must feed your horses for at least 10 minutes every 20 minutes, and your cattle for at least 25 minutes every 50 minutes. The period for the horse is 20 minutes, and the execution time is 10 minutes. The period for the cattle is 50 minutes, and the execution time is 25 minutes. The deadline for each matches the period. They eat in the same barn out of the same troughs so they cannot ever eat at the same time. Assume the required times include the time it takes to change from feeding horses to cows and back each time.

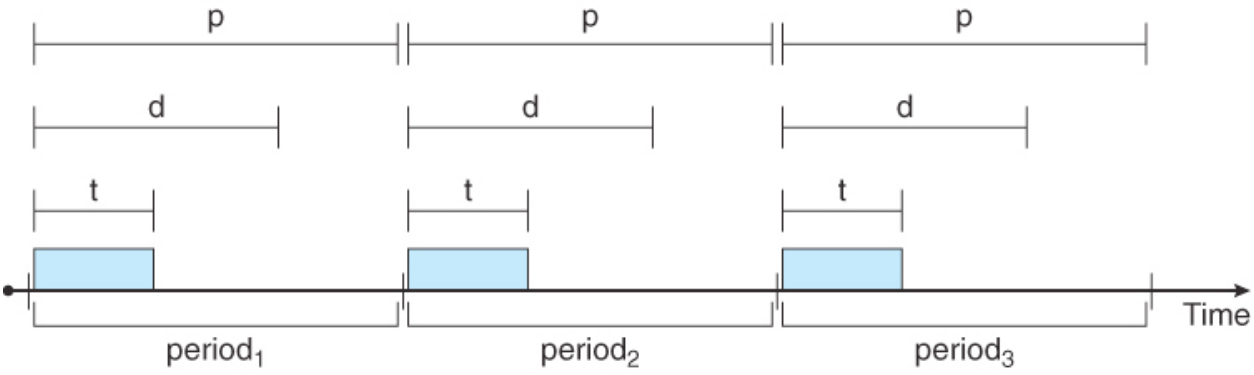
If you take turns feeding the cattle and then the horses, then the horses would starve to death before they are fed and so this is not a valid scheduling order:



If, on the other hand, you were to feed them in the following order, then it is possible to satisfy the given constraints for the full 100 minutes. Note that the LCM of 20 and 50 is 100. For this example, both the horse and the cow had deadlines that corresponded to their period. That is, as long as the horse and the cow were fed for 10 and 25 minutes every 20 and 50 minutes, respectively, then everything was fine. Note the last 30 minutes below could also have been Horse - 10, Cow - 10, Horse - 10 and still met the given constraints.



Not all tasks have deadlines that correspond to their periods though. In this next example, a user might click a button every 60 seconds expecting an on-screen response within 3 seconds of the click, but the response may take 1 second for the computer to compute. In this case, from the moment the button is clicked to start the task until it is clicked again to start the next task, the period is 60 seconds, the deadline is 3 seconds, and the execution time is 1 second.



Input

The first line contains a single integer N, the number of test cases to follow. The first line of each test case contains a single integer M, the number of tasks to schedule. The following N lines each represent a single task, made up of three integers in the form P D T, where P is the period with which the task repeats, D is the deadline with which the task must be completed each time it repeats, and T is the time required to execute the task. For instance, a button which will be clicked every 60 seconds, expecting an on-screen response within 3 seconds, and the response taking 1 second to compute, would be written as '60 3 1'.

Output
For each set of periodic tasks, determine if there exists some scheduling order such that the constraints can always be met. Print SCHEDULABLE if some order exists, or print NOT SCHEDULABLE if no order exists.

Constraints
1 <= N <= 10
1 <= M <= 8
1 <= LCM <= 1024
1 <= T <= D <= P <= 1024 for all tasks

Sample Input File
```
4
2
20 20 10
50 50 25
2
3 3 1
4 4 2
2
2 1 1
2 2 1
2
2 1 1
2 1 1
```

Example Output to Screen
```
SCHEDULABLE
SCHEDULABLE
SCHEDULABLE
NOT SCHEDULABLE
```

Explanation of Output

In the first set of tasks, there are two tasks corresponding to the rancher example above. Since, as per the diagrams, we know that there is a solution, the result is SCHEDULABLE.

In the second set of tasks, there are two tasks. The first task appears every 3 seconds, and within 3 seconds of appearing must be executed for at least 1 second. The second task appears every 4 seconds, and within 4 seconds of appearing must be executed for at least 2 seconds. The LCM of 4 and 3 is 12. The following is a valid scheduling of these two tasks:
Time 0 – 1: Task 1
Time 1 – 3: Task 2
Time 3 – 4: Task 1
Time 4 – 6: Task 2
Time 6 – 7: Task 1
Time 7 – 8: Task 1 or 2
Time 8 – 10: Task 2
Time 10 – 11: Task 1
Time 11 – 12: Task 1 or 2

In the third set of tasks, there are two tasks. The first task appears every 2 seconds, and within 1 second of appearing must be executed for at least 1 second. The second task appears every 2 seconds, and within 2 seconds of appearing

must be executed for at least 1 second. The LCM of 2 and 2 is 2. The following is a valid scheduling of these two tasks:
Time 0 – 1: Task 1
Time 1 – 2: Task 2

The fourth set of tasks is not schedulable. Both task 1 and task 2 must be executed for at least 1 second, within 1 second of appearing. Since both tasks appear at the beginning and we can only execute one of them at a time for one second, the other task will be overdue.